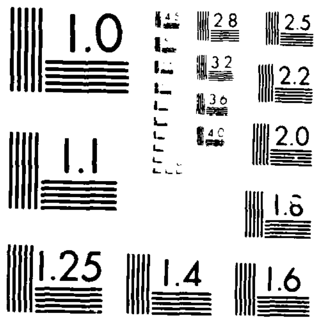
AD-A164 795 SAFETY WITHOUT STUTTERING(U) CORNELL UNIV ITHACA NY DEPT OF COMPUTER SCIENCE B ALPERN ET AL. OCT 85 CU-CSD-TR-85-708 N00014-86-K-0092 1/1

UNCLASSIFIED F/G 9/2 NL

END

FILMED

DTIC




MICROCOPY RESOLUTION TEST CHART

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

(12)

# REPORT DOCUMENTATION PAGE

AD-A164 795

| | |
|---|---|
| 1a REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b RESTRICTIVE MARKINGS |
| 2a SECURITY CLASSIFICATION AUTHORITY | 3 DISTRIBUTION/AVAILABILITY OF REPORT<br>Unlimited |
| 2b DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4 PERFORMING ORGANIZATION REPORT NUMBER(S)<br>Cornell University TR 85-708 | 5 MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a NAME OF PERFORMING ORGANIZATION<br>Cornell University / 6b OFFICE SYMBOL (If applicable) | 7a NAME OF MONITORING ORGANIZATION<br>Office of Naval Research |
| 6c ADDRESS (City, State, and ZIP Code)<br>Dept. of Computer Science<br>Cornell University<br>Ithaca, NY 14853 | 7b ADDRESS (City, State, and ZIP Code)<br>800 North Quincy Street<br>Arlington, VA 22217-5000 |
| 8a NAME OF FUNDING/SPONSORING ORGANIZATION<br>Office of Naval Research / 8b OFFICE SYMBOL (If applicable) | 9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>N00014-86-K-0092 |
| 8c ADDRESS (City, State, and ZIP Code)<br>800 North Quincy Street<br>Arlington, VA 22217-5000 | 10 SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO |

11 TITLE (Include Security Classification)

Safety without Stuttering

12 PERSONAL AUTHOR(S)

Bowen Alpern, Alan J. Demers, Fred B. Schneider

| 13a TYPE OF REPORT | 13b TIME COVERED | 14 DATE OF REPORT (Year, Month, Day) | 15 PAGE COUNT |
|---|---|---|---|
| interim | FROM ____ TO ____ | October 1985 | 4 |

16 SUPPLEMENTARY NOTATION

| 17 COSATI CODES | | | 18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | safety, invariance under stuttering, temporal logic, concurrent program's, program properties |

19 ABSTRACT (Continue on reverse if necessary and identify by block number)


A new formalization of safety properties is given. The formalization agrees with the informal definition - that a safety property stipulates that some "bad thing" doesn't happen during execution - for properties that are not invariant under stuttering, as well as for properties that are.


DTIC FILE COPY



86 2 27 080

| | |
|---|---|
| 20 DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | 21 ABSTRACT SECURITY CLASSIFICATION |
| 22a NAME OF RESPONSIBLE INDIVIDUAL<br>Fred B. Schneider | 22b TELEPHONE (Include Area Code)<br>607-255-9221 / 22c OFFICE SYMBOL |

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

# Safety without Stuttering*

Bowen Alpern
Alan J. Demers[+]
Fred B. Schneider

TR 85-708
October 1985

Department of Computer Science
Cornell University
Ithaca, NY 14853

* This work is supported, in part, by NSF Grant DCR-8320274 and a grant from the Office of Naval Research.

[+] Current address: Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304.

# Safety without Stuttering*

Bowen Alpern
Alan J. Demers
Fred B. Schneider

Department of Computer Science
Cornell University
Ithaca, New York 14853

October 22, 1985

ABSTRACT

A new formalization of safety properties is given. The formalization agrees with the informal definition—that a safety property stipulates that some "bad thing" doesn't happen during execution—for properties that are not invariant under stuttering, as well as for properties that are.

*This work is supported, in part, by NSF Grant DCR-8320274 and a grant from the Office of Naval Research.

## 1. Introduction

Informally, a safety property stipulates that some "bad thing" doesn't happen during execution [Lamport 77]. Examples of safety properties include mutual exclusion, deadlock freedom, and partial correctness. In *mutual exclusion*, the proscribed "bad thing" is two processes executing in critical sections at the same time. In *deadlock freedom*, it is deadlock. In *partial correctness*, it is terminating in a state not satisfying the postcondition when execution is started in a state that satisfies the precondition.

A formal definition of safety is given in [Lamport 85]. While that definition correctly captures the intuition for an important class of properties—those invariant under stuttering—it is inadequate for safety properties that are not invariant under stuttering. This note gives a formal definition of safety that is independent of stuttering.

Section 2 of the paper reviews some notation for describing properties. Section 3 gives our new formalization of safety and relates it to the one in [Lamport 85]. Finally, section 4 puts our work into perspective.

## 2. Properties

An execution of a concurrent program can be described by an infinite sequence of states

$$\sigma = s_0 s_1 \ldots$$

which we call a *history*. Each state after $s_0$ results from executing a single atomic action in the preceding state. For a terminating program execution, an infinite sequence is obtained by repeating the final state. This corresponds to the view that a terminating execution is the same as a non-terminating execution in which after some finite time (once the program has terminated) the state remains fixed.

A *property* is a set of histories. We write $\sigma \models P$ to denote that history $\sigma$ is a member of property $P$. A property is usually defined by a characteristic predicate on histories rather than by enumerating the histories themselves. Temporal logic provides a suitable formalism for this purpose [Lamport 83].

The following notation is used in the remainder of the paper. $S$ is the set of states, $S^*$ the set of finite sequences of states, and $S^\omega$ the set of histories. For a history $\sigma = s_0 s_1 \ldots$, define

$$\sigma[i] \;=\; s_i$$

$$\sigma[..i] \;=\; s_0 s_1 \ldots s_i$$

$$\sigma[i..] \;=\; s_i s_{i+1} \ldots$$

We use superscripts to denote repetition. Thus, for $\alpha$ in $S^*$, $\alpha^n$ is the finite sequence obtained by repeating $\alpha$ $n$ times and $\alpha^\omega$ is the history obtained by repeating $\alpha$ indefinitely. We use juxtaposition to denote catenation of state sequences.

### 3. Formalizing Safety

If a "bad thing" happens in a history, then it must do so in some finite prefix of that history. Based on this, Lamport [Lamport 85] formalized a safety property as any property $P$ satisfying

$$SP_L(P):\quad (\forall\sigma\colon\ \sigma\in S^\omega\colon\ \sigma\models P \Leftrightarrow (\forall i\colon\ 0\leq i\colon\ \sigma[..i]\sigma[i]^\omega\models P))$$

Thus, a safety property $P$ is satisfied by a history $\sigma$ if and only if every prefix of $\sigma$—extended to an infinite sequence by repeating its last state—also satisfies $P$. Extension of a finite sequence ($\sigma[..i]$) to an infinite one is necessary because only a history can satisfy a property; repetition of the last step is one of a number of ways to perform this extension.

For some properties, extending a finite sequence by repeating the final state causes problems. Consider property $CP$ stipulating that a variable *clock* is increased for every instruction executed. Using the temporal logic notation "$\bigcirc$" for the "next-time" operator, this is given by

$$CP:\quad (clock=N) \Rightarrow \bigcirc(clock>N).$$

Intuitively, $CP$ is a safety property: the "bad thing" is *clock* not increasing in two successive states. However, $CP$ does not satisfy the formal definition of safety given above. $SP_L(CP)=false$ because for no history $\sigma$—even if $\sigma\models CP$—will the value of *clock* change after the $i^{th}$ state in $\sigma[..i]\sigma[i]^\omega$.

This difficulty arises only for properties that are not invariant under stuttering. A property is *invariant under stuttering* if and only if whenever a history satisfies the property, the history with every state repeated zero or more times also satisfies the property, and vice versa. More formally, any property $P$ satisfying

$$STR(P):\quad (\forall f\colon\ f\in \mathrm{N}\to\mathrm{N}\colon\ \sigma\models P \Leftrightarrow \sigma[0]^{f(0)+1}\ldots\sigma[i]^{f(i)+1}\ldots\models P)$$

is invariant under stuttering. Properties that are invariant under stuttering are well suited for hierarchical specification and verification [Lamport 83]. By permitting states to be repeated, meaningful statements can be made about the system at various levels of abstraction. For example, execution of a higher-level operation that is implemented by a sequence of lower-level operations can be viewed as a sequence of repeated, identical, higher-level states where there is one state for every lower-level instruction executed but the last, which produces a new higher-level state.

We now give a formalization of safety that agrees with $SP_L$ for properties invariant under stuttering and that agrees with the informal definition of safety for properties (like $CP$)

that are not. If a safety property $P$ does not hold for a history $\sigma$, then some "bad thing" must have happened during $\sigma$. This "bad thing" must be irremediable, because a safety property requires that the "bad thing" never happen. Thus, if $\neg(\sigma \models P)$, there is some prefix of $\sigma$ (that includes the "bad thing") for which no extension to a history will satisfy $P$. Taking the contrapositive of this, $P$ is a safety property if it satisfies

$$SP_{ADS}(P): \ (\forall \sigma: \ \sigma \in S^{\omega}: \ \sigma \models P \Leftrightarrow (\forall i: \ 0 \le i: \ (\exists \beta: \ \beta \in S^{\omega}: \ \sigma[..i]\beta \models P))).$$

$SP_{ADS}$ differs from $SP_L$ in the way prefixes are extended to form histories. $SP_{ADS}$ permits extension using any history $\beta$, while $SP_L$ requires extension by replicating the last state of the prefix. Note that $SP_{ADS}(CP) = true$, so $CP$ is a safety property according to this formalization.

The relationship between $SP_L$ and $SP_{ADS}$ is given in the following two theorems. The first theorem states that safety properties under $SP_L$ are also safety properties under $SP_{ADS}$.

**Theorem:** For any property $P$, $SP_L(P) \Rightarrow SP_{ADS}(P)$.

**Proof:** Assuming $SP_L(P)$, we must show $\sigma \models P \Leftrightarrow (\forall i: \ 0 \le i: \ (\exists \beta: \ \beta \in S^{\omega}: \ \sigma[..i]\beta \models P))$.

$$
\begin{aligned}
& (\forall i: \ 0 \le i: \ (\exists \beta: \ \beta \in S^{\omega}: \ \sigma[..i]\beta \models P)) \\
\Leftrightarrow\ & (\forall i: \ 0 \le i: \ (\exists \beta: \ \beta \in S^{\omega}: \ \sigma[..i]\sigma[i]^{\omega} \models P)) \quad SP_L(P), \text{ since } \sigma[..i]\beta \models P \\
\Leftrightarrow\ & (\forall i: \ 0 \le i: \ \sigma[..i]\sigma[i]^{\omega} \models P) \quad \text{by Predicate Logic} \\
\Leftrightarrow\ & \sigma \models P \quad \text{by } SP_L(P).
\end{aligned}
$$

□

The second theorem states that every safety property according to $SP_{ADS}$ that is invariant under stuttering is also a safety property according to $SP_L$.

**Theorem:** For any property $P$, $(SP_{ADS}(P) \wedge STR(P)) \Rightarrow SP_L(P)$.

**Proof:** Assuming $SP_{ADS}(P)$ and $STR(P)$, we must show:

(1) $\sigma \models P \Rightarrow (\forall i: \ 0 \le i: \ \sigma[..i]\sigma[i]^{\omega} \models P)$

(2) $(\forall i: \ 0 \le i: \ \sigma[..i]\sigma[i]^{\omega} \models P) \Rightarrow \sigma \models P$

First, we prove (1):

$$
\begin{aligned}
& \sigma \models P \\
(*)\ \Leftrightarrow\ & (\forall i: \ 0 \le i: \ (\exists \beta: \ \beta \in S^{\omega}: \ \sigma[..i]\beta \models P)) \quad \text{by } SP_{ADS}(P) \\
\Leftrightarrow\ & (\forall i: \ 0 \le i: \ (\exists \beta: \ \beta \in S^{\omega}: \ (\forall n: \ 0 \le n: \ \sigma[..i]\sigma[i]^{n}\beta \models P))) \quad \text{by } STR(P) \\
\Rightarrow\ & (\forall i: \ 0 \le i: \ (\forall n: \ 0 \le n: \ (\exists \beta: \ \beta \in S^{\omega}: \ \sigma[..i]\sigma[i]^{n}\beta \models P))) \quad \text{by Predicate Logic} \\
\Leftrightarrow\ & (\forall i: \ 0 \le i: \ (\forall n: \ 0 \le n: \ (\exists \beta: \ \beta \in S^{\omega}: \ (\sigma[..i]\sigma[i]^{\omega})[..i+n]\beta \models P))) \\
& \qquad \text{since } \sigma[..i]\sigma[i]^{n} = (\sigma[..i]\sigma[i]^{\omega})[..i+n] \\
\Leftrightarrow\ & (\forall i: \ 0 \le i: \ (\forall j: \ i \le j: \ (\exists \beta: \ \beta \in S^{\omega}: \ (\sigma[..i]\sigma[i]^{\omega})[..j]\beta \models P))) \quad \text{by Predicate Logic} \\
\Leftrightarrow\ & (\forall i: \ 0 \le i: \ (\forall j: \ 0 \le j: \ (\exists \beta: \ \beta \in S^{\omega}: \ (\sigma[..i]\sigma[i]^{\omega})[..j]\beta \models P))) \\
& \qquad \text{since } j < i \Rightarrow (\sigma[..i]\sigma[i]^{\omega})[..j] = \sigma[..j] \text{ and according to } (*),\ \sigma[..j]\beta \models P \\
\Leftrightarrow\ & (\forall i: \ 0 \le i: \ \sigma[..i]\sigma[i]^{\omega} \models P) \quad \text{since } SP_{ADS}(P).
\end{aligned}
$$

Next, we prove (2):

$$
\begin{array}{ll}
 & (\forall i:\ 0 \le i:\ \sigma[..i]\sigma[i]^{\omega} \models P) \\
\Rightarrow & (\forall i:\ 0 \le i:\ (\exists \beta:\ \beta \in S^{\omega}:\ \sigma[..i]\beta \models P)) \quad \text{use } \beta = \sigma[i]^{\omega} \\
\Leftrightarrow & \sigma \models P \quad \text{by } SP_{ADS}(P).
\end{array}
$$

□

## 4. Discussion

It has been argued that properties invariant under stuttering are the only ones of real interest in program verification [Lamport 83]. We agree. This, however, is a religious issue. A formalization of safety should serve many faiths. This note presents a definition of safety that can be applied to any property.

**Acknowledgments**

Suggestions by D. Gries led to improvements in the clarity of the proofs.

**References**

[Lamport 77] Lamport, L. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering SE-3*,2 (March 1977), 124-143.

[Lamport 83] Lamport, L. What good is temporal logic? *Information Processing 83*, R.E.A Mason, ed., (1983), North Holland, Amsterdam.

[Lamport 85] Lamport, L. Logical Foundation. In *Distributed Systems—Methods and Tools for Specification*, Lecture Notes in Computer Science, Vol 190. M. Paul and H.J. Siegert, eds. (1985), Springer-Verlag, New York.

END

FILMED

4-86

DTIC